A Component Approach to Object Model Standards for Simulation

Major Leroy A. Jackson
Operations Research Analyst
US Army TRADOC Analysis Center—Monterey
(408) 656-4061
jacksonl@mtry.trac.nps.navy.mil

Summary. Object models are an important feature of the United States Department of Defense (DoD) High Level Architecture (HLA) and the Defense Modeling and Simulation Office (DMSO) Conceptual Model of the Mission Space (CMMS). Currently, all major DoD simulations under development use object-oriented methodologies. The major benefits of object-oriented programming include software reuse, improved maintainability, interoperability, and rapid prototyping. A set of standard objects is needed to establish consistency among future Army models and simulations. This paper describes a component approach proposed for object model standards development.

1. INTRODUCTION

This paper describes a component approach for object-oriented modeling and design which has been adopted for standards development in the US Army modeling and simulation community. This design approach directly supports the goals for developing object modeling standards by fostering model reuse and improving model interoperability.

2. BACKGROUND

In May 1997, the US Army Training and Doctrine Command (TRADOC) Analysis Center (TRAC) in Monterey, California (TRAC—Monterey) began a study sponsored by the Army Modeling and Simulation Office (AMSO) to support standards development for Army modeling and simulation objects. [1] The study team was led by a military analyst at TRAC—Monterey and included a professor and two graduate students from the Operations Research Department of the Naval Postgraduate School. The study advisory group included senior analysts from the major Army analytical agencies. The team examined selected models from existing and future simulations under development in order to provide examples and insights to support object standards development. The team also developed an approach to object model standards development, drafted sample standards for platforms (entities) and units, and drafted sample guidelines for the use of standard objects. The study team determined that object model standards would focus on high-level abstract classes containing a minimal, essential set of class methods. Rather than specify standard attributes for classes, *get* and *set* methods would signify the data content of standard objects. An important aspect of the study team recommendations was the component approach to object model standards.

3. APPROACHES TO REUSE

The two main approaches to reuse in object oriented designs are class inheritance and object composition. [2&3] Each approach has distinct advantages and disadvantages.

3.1 Inheritance

Inheritance allows subclasses to extend and specialize a parent class by adding data and methods, and by replacing the method implementation of the parent class with a new implementation. Inheritance is straightforward since it is directly supported by object oriented languages. General classes are placed higher in the inheritance hierarchy and more specialized objects lower, so several subclasses may reuse the parent class. Inheritance, however, breaks encapsulation by exposing the parent class implementation to its subclasses. Implementation changes in the parent class often necessitate changes in subclasses. Issues of multiple inheritance and the requirement for compile-time binding further dilute the value of inheritance for reuse. Inheritance promotes implementation dependencies. Despite some minor disadvantages, inheritance is an extremely important feature in object oriented systems. Inheritance of abstract classes provides common protocols or interfaces in an object-oriented design. This technique ameliorates some of the pitfalls in the use of inheritance.

3.2 Object Composition

Object composition is the construction of a class using instances of other classes as components. Because component classes are accessed through their interface (public methods), encapsulation is not broken and there are significantly fewer implementation dependencies. Object composition is, however, more difficult. It requires that component classes have well defined interfaces that promote reuse. In addition, objects must respect these interfaces since no implementation details are exposed. Finally, object composition proliferates numerous small component classes since each component class must focus on relatively few tasks. This often requires many interrelationships among the component classes that would normally be encapsulated in one larger class.

3.3 The Component Approach to Standards

The component approach to standards favors object composition over class inheritance, but exploits the advantages of both approaches. With the component approach, classes of interest are constructed by selecting and implementing abstract component classes. Component classes are implemented and possibly extended through inheritance. The principle advantage of the component approach to standards over alternative approaches is it focuses on the development of standard interfaces rather than the construction of a single monolithic class hierarchy. If a single class interface supports several different implementation schemes, then the goal of "plug and play" software components is achieved. For example, if the same method signature (set of parameters required to invoke the method) supports several attrition schemes (Lanchester, Bonder-Ferrel etc.) then it is possible to substitute one attrition algorithm for another without making other changes in the simulation.

4. STANDARD M&S OBJECTS

This section provides examples of standard modeling and simulation (M&S) objects developed using the component approach and discusses the problem of determining the appropriate level of detail for standards using the component approach.

4.1 *Location* Class Example

The notion of location is fundamental to most military simulations. There are numerous coordinate systems used in simulation; each is appropriate for some simulations and not suitable for others. A common, abstract location object can foster interoperability among simulations that use different coordinate schemes. In this example (see next page), the *Location* class abstracts the concept of location by providing a method to calculate the distance between locations and to convert to an unspecified standard location scheme. The *Location* class has two standard subclasses, *Local* and *Geocentric*, which illustrate the two main competing coordinate schemes. Each provides location through *get* methods. [4] The *Location* class is powerful and flexible. Suppose one has a simulation that uses a network of arcs and nodes. The distance between nodes is stored in a table and the distance from a node along an arc is calculated based on the fraction of the arc traversed at the time a distance is requested. The simulation developer conforms to the standard by simply subclassing the *Location* class and implementing its methods.

```
                         Location
                 -------------------------
                 -------------------------
                      distanceFrom()
                         convert()
      Local                                              Geocentric
-------------------------                         -------------------------
-------------------------                         -------------------------
  getXCoordinate                                      getLattitude
  getYCoordinate                                      getLongitude
  getZCoordinate                                      getAltitude
```

Location Class Hierarchy

4.2 *PlatformComponent* Example

Entity level simulations of combat generally have a notion of platform or entity upon which most militarily significant actors from individual combatants to tanks to aircraft are based. While the details vary significantly among various simulations, there are common aspects of all platforms in almost all entity level simulations. The standard platform components are *Location*, *Communications*, *Movement*, *Sensor*, *Weapon*, *Carrier*, *Crew*, *PlatformFrame* and *Logistics* (with *Supply* and *Maintenance* subclasses). These components are subclasses of the *PlatformComponent* class that provides *getType* and *getStatus* methods to all components. (The interested reader can refer to [4,5&9] for the details of the platform components.) A simulation developer composes platforms in an entity-level simulation using zero or more of each of components as appropriate. Implementation details are left to the developer, but each component provides a standard interface into a significant aspect of the entity as illustrated by the *Location* class described above. The standard platform components are flexible. The simulation developer uses only the components required in the simulation. If, for example, the crew is not modeled, then that component is omitted. There is no restriction on the number or type of weapons, sensors or communications systems on the platform.

4.3 Levels of Detail for Standards

The component approach does not solve the problem of determining the appropriate level of detail for standard classes, but it provides a suitable context for debate on this issue. The study team used several general rules to determine if a method belonged in a standard class. The primary rule was that the method be essential to support a function found in almost all simulations where the component would be found. The study team made a conscious effort to err on the side of proposing minimal standards to avoid creating a large burden for the simulation developer. The shared vision was of abstract components as the basis for standards. In the approach described, the abstract components are sufficient to assemble a platform that represents the abstract tank. Further refinement would be required to produce a generic tank and still more refinement to produce a detailed model of an actual tank. Each level is a possible standard, but the fraction of simulations which might support the more detailed standards is rather small.

5. CONCLUSION

The US Army modeling and simulation community is reviewing standard component models for platform and unit objects which evolved from the study. The Object Management Standards Coordinating Committee has proposed a general framework for object model development and is actively developing standard component models for a variety of other significant objects found in ground combat simulations. The component approach to object modeling promotes reuse of models and improves model interoperability. It focuses on the development of a standard object interface which consists of the minimum, essential set of abstract class methods in a component.

6. ACKNOWLEDEMENTS

This work was sponsored by the Deputy Undersecretary of the Army for Operations Research through the auspices of the US Army Modeling and Simulation Office. I am particularly indebted to Professor Arnold Buss of the Naval Postgraduate School for his keen insights and tremendous contributions to the study.

7. ABOUT THE AUTHOR

Major Leroy A. Jackson is an Army officer with over twenty years of enlisted and commissioned service. He graduated with a BA in Mathematics from Cameron University in 1990 and with an MS in Operations Research from the Naval Postgraduate School in 1995. He is currently an operations research analyst at the U.S. Army Training and Doctrine Command (TRADOC) Analysis Center (TRAC) Research Activities in Monterey, California and continues graduate studies in operations research at the Naval Postgraduate School.

8. REFERENCES

[1] Jackson, Leroy A. (April 1997) *Standard Army M&S Objects Study Plan*, US Army TRADOC Analysis Center—Monterey.
[2] Gamma, Erich, Richard Helm, Ralph Johnson and John Vlissides (1995), *Design Patterns: Elements of Reusable Object-Oriented Software Reuse*, Addison-Wesley.
[3] Jacobson, Ivar, Magnus Christerson, Patrik Jonsson and Gunnar Overgaard (1995), *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley.
[4] Buss, Arnold, and Leroy Jackson (September 1997), *Standard Army Modeling and Simulation Objects: Interim Report*, US Army TRADOC Analysis Center— Monterey.
[5] Dudgeon, Douglas E. (September 1997) *Development a Standard Platform-Level Army Object Model*, MS Thesis, Department of Operations Research, Naval Postgraduate School.
[6] Cotton, Arthur L. III (September 1997) *Developing a Standard Unit-Level Object Model*, MS Thesis, Department of Operations Research, Naval Postgraduate School.